1 Introduction

First, original ers code had been edited, so that only needed information is printed out in the form: “parameters: fields/rule %d and rules/class %d; Training accuracy %d; Test accuracy %d”. Second, a separate small C program had been written that spawns child processes, running instance of ers with specified parameters. Third, a separate Python program had been written that uses Numpy library for dataset file manipulation and generation of new datasets. In particular, written Python script creates copy of the original testing dataset, selects random line, adds it to the training dataset and removes that line from the testing dataset (moving 900/2000 lines from testing dataset to training dataset). Source code at: http://www2.macs.hw.ac.uk/~bm4/DMML/CW3/

2 Results

Two sets of experiments were conducted: 1) the variation between the two parameters\(^1\) is small — both parameters are increased gradually for every experiment; 2) the variation between the two parameters high in the beginning and the end of experiment, while in the middle variance between the two becomes smaller. This is done to see how parameter decoupling affects performance.

![Figure 1: Low variance: Training performance](image1)

![Figure 2: Low variance: Testing performance](image2)

Low variance between number of fields/rule and number of rules/class on the edge of the spectrum.

![Figure 3: High variance: Training performance](image3)

![Figure 4: High variance: Testing performance](image4)

High variance between number of fields/rule and number of rules/class on the edge of the spectrum.

\[^1\] Fields per rule and rules per class (in this order)
3 Variations in Performance

All the variations had been measured in combination with both parameters as it is stated in the task description (“Evaluate the performance of ers.c for 16 combinations of fields-per-rule and rules-per-class”)

3.1 Variation in performance with number of fields per rule

As it can be seen from the graphs provided above, in case of both low and high variability between the parameters, as the number of fields per rule increases, the testing performance decreases, whereas the training performance increases in case of low variability and decreases in case of high variability. The same applies to all datasets tested.

This indicates that during testing, it is desirable to use low variance between the parameters and use more fields per rule to achieve better performance. Unfortunately, this will not be the case during the testing phase.

In addition, using low variability for testing achieves half of the performance as compared to the testing using high variability.

3.2 Variation in performance with number of rules per class

Considering combinations, as the number of rules per class increases in the case of testing, the performance decreases in both low and high variance between the parameters.

In the case of training phase, the performance improves for the low variance and the performance drops down for the high variance as the number of rules per class is increased.

3.3 Variation in performance with size of training set

For the low variance between the parameters, the performance is worse in the beginning (between parameters being 1/1 and 25/19), but then the performance ‘catches up’ with the performance on the original dataset. It is worth noting that the performance becomes worse on mentioned interval as more instances are used during the training phase.

For the high variance between the parameters, the performance actually becomes better (after the parameters being 17/32 and onwards) as 900 instances are used during training phase. The performance eventually becomes worse as more more instances (2000) are used during testing phase.

It is important to note that although the performance becomes better with 900 instances used for training, the training performance gradually drops.

In general, as the size of the training set increases, performance worsens with a single exception described above. In this particular analysis, as the number of parameters increases, the more pixels are considered for rule generation as well as more rules are generated for every handwritten digit, which is a good thing. With that being said, since the digits are handwritten and less likely to be described by a set of pixels (as the position, style of writing are different for every person),
the more parameters are used for rule generation, the less likely they will be able to cover the ‘general case’, whereas the less rules are used with very few parameters, the more likely they fit all the handwritten digit, but less likely to describe one particular handwritten digit. That is why the performance behaviour is as shown on the diagrams — high in the beginning (less parameters fit the most handwritten digits) and low at the end (more parameters fit very few handwritten digits).